COMPARISON OF ADAPTIVE NEURO-FUZZY AND PARTICLE SWARM OPTIMIZATION BASED NEURAL NETWORK MODELS FOR FINANCIAL TIME SERIES PREDICTION

In this paper, an attempt has been made to assess the forecasting ability of adaptive neuro-fuzzy inference system (ANFIS) with the traditional feed forward neural network using financial time series data. Also, efforts have been made to examine the performance of particle swarm optimization algorithm for training neural networks. This algorithm is shown to perform well in the current study.

Introduction

Time-series forecasting is one of the challenging areas of forecasting in which past observations of a given data set are used to build a model for projecting the future values of the data set. In recent years, the analysis and forecasting of financial time series has attracted a lot of attention of researchers. Financial series, such as returns on stock and exchange rate, are inherently noisy, non-stationary and leptokurtic, hence it is difficult to capture the future behaviour on the basis of past records. Much effort has been devoted over the past several decades to develop and improve time-series forecasting. Initially, linear models such as the Box-Jenkins or ARIMA method (Box & Jenkins, 1976) were used for time series prediction. These models assume that the underlying mechanism for the data generation process is linear in nature. However, most of the real life problems are often non-linear. Several time-series models with non-linear structure such as the bilinear model (Granger & Anderson, 1978), the threshold autoregressive (TAR) model (Tong, 1990) were developed in the past. These non-linear models can provide improved point forecast for the observed data provided the true model is known. Other non-linear models such as the auto regressive conditional heteroscedasticity (ARCH) (Engle, 1982) and the generalized ARCH (GARCH) (Bollerslev, 1986) modelled the changes in variance (or volatility) enabling better assessment of risk by providing better estimates of the variance (Chatfield, 1996). However, the major limitation of all these non-linear models relies on knowing explicit relationships of the data series which is generally unpredictable.

Artificial neural networks (ANNs) can be a potential tool for non-linear processes that have unknown relationship and as a result are difficult to fit (Darbellay & Slama 2000). ANNs are non-linear, data-driven and self adaptive approaches as opposed to the above model-based non-linear methods. One of the major application areas of ANNs is forecasting (Zhang, Patuwo, & Hu, 1998). ANN can identify and learn correlated patterns between input data sets and corresponding target values. This technique is

---

1 The first author would like to acknowledge the financial support from Department of Science and Technology, Government of India for his visit to University of Manitoba, Canada; the last two authors acknowledge partial financial support from Natural Sciences and Engineering Research Council of Canada.
ideally suited for modeling imprecise and noisy data which is a desirable feature for financial time series modelling. Therefore, ANN is a more general and flexible modeling technique for time series forecasting. However, training of a neural network model is complicated due to its non-linearity for optimization and large amount of computational time requirement. The standard back propagation training algorithm (Rumelhart et al, 1986) for neural networks exhibits slow convergence, local minima, and lack of robustness. Essentially, at present there does not exist any algorithm which can guarantee the global optimal solution for a general non-linear optimization problem in a stipulated time (Zhang, 2007). An efficient training algorithm should provide better estimation for neural network model. In view of the above, several variants or modification of back propagation algorithm have been proposed in the literature. Recently, evolutionary computing techniques (Fogel, 1998; Kennedy, Eberhart, & Shi, 2001) have been used in neural networks. One particular technique, Particle Swarm Optimization (PSO) (Kennedy, Eberhart, & Shi, 2001; Brabazon and O’neil, 2006; Clerc, 2006) has been used in training neural networks and more recently in evolving neural network architecture.

PSO is one of the computational intelligence tool developed by Kennedy and Eberhart in 1995 (Kennedy & Eberhart, 1995). It is a nature inspired population based optimization algorithm which uses collective intelligence emerging from the cooperation of individual members within a social system. PSO has been regarded as having many similarities with genetic algorithms (GA) while the former is generally categorized as nature-inspired algorithm and the latter as evolutionary algorithm. Both PSO and GA initialize the system with a set of random populations and try to find the optimum solution in the search space. However, the crossover and mutation operators that are central to genetic algorithms do not exist in PSO. Memory is important in PSO. GA depends on Darwin's theory, survival of the fittest. Therefore, the population decreases, by eliminating the weakest in the solution. Genetic algorithms share information through the chromosomes. In PSO, information is shared using the two best values only. This allows all particles to converge to the best solution. In this paper, we use PSO for training neural networks. PSO based neural networks have been used for solving scientific and engineering problems like diagnosis of human tremor, electric power and voltage management, and computer numerical control machine optimization (Kennedy, Eberhart, & Shi, 2001); the technique has not been explored much in the area of time series prediction. To our knowledge, the only other study (Chen, Yang, & Dong, 2006) discusses a hybrid training algorithm involving PSO for training the local linear wavelet neural network for time series prediction.

Further, neural networks has the ability of self learning and non-linear approximations but it lacks the interpretation capability, i.e., they are unable to explain about a particular decision to the user in a human-comprehensible form. The lack of explanatory capability of neural network can be improved by combining with another complimentary technology known as fuzzy logic. Fuzzy rule-based models are easy to comprehend because it uses linguistic terms and the structure of if-then rules. Unlike neural network, fuzzy logic does not come with a learning algorithm. Since neural networks can learn, it is natural to combine the two technologies to achieve better results. A neuro-fuzzy system can be loosely defined as a system that uses a judicious combination of the merits of neural and fuzzy approaches. Neuro-fuzzy hybridization results in a hybrid intelligent system that synergizes these two techniques by combining the human-like reasoning style of fuzzy systems with the learning and connectionist structure of neural networks. Jang (Jang, 1993; Jang, 1997) developed an architecture and learning procedure of a neuro-fuzzy system known as ANFIS (adaptive network-based fuzzy inference system or semantically equivalently adaptive neuro fuzzy inference system) and demonstrated its applications to automatic control and signal processing.

In this paper, we compared the prediction performances of ANFIS, PSO based neural networks and a standard back-propagation (Levenberg-Marquardt algorithm) based neural networks. Models were developed using daily closing price data of S&P 500 index. The performance of these techniques has been
evaluated with respect to the mean squared error (MSE) and the mean absolute error (MSE) for three forecasting horizons.

**Neural Networks for Time Series Forecasting**

Neural networks are computational methods that mimic the behaviour of the human brain's central nervous system. They are considered as a class of generalized non-linear, non-parametric, data driven statistical methods (White, 1989). In neural network architecture, input layer is separated from output layer by one or more hidden layer. Each layer contains one or more nodes. Essentially input variables are getting filtered through hidden layer(s) before reaching to output layer. Non-linearity structure of the data is being mapped through hidden layer(s). There are many different types of neural networks architectures, but the most common and successful neural network for time series forecasting is multi-layer feed forward neural network (Zhang, Patuwo, & Hu, 1998), which is used for the present study. All the layers in a multi-layer feed forward neural network are connected through an acyclic arc.

The neural network structure for a particular problem in time series prediction includes determination of number of layers and total number of nodes in each layer. This is usually determined through experimentation of the given data as there is no theoretical basis for determining these parameters. It has been proved that neural networks with one hidden layer can approximate any non-linear function given sufficient number of nodes at hidden layer and adequate data points for training (Cybenko, 1989). In this study, we have used neural network with one hidden layer. In time series analysis, the determination of number of input nodes which are lagged observations of the same variable plays a crucial role as it helps in modelling the autocorrelation structure of the data (Zhang, 2003). The determination of number of output nodes is relatively easy. In this paper, one output node has been used and multi-step ahead forecasting has been done using the iterative procedure as used in Box-Jenkins method. This involves use of forecast value as an input for forecasting the future value. It is always better to select the model with small number of nodes at hidden layer as it improves the out-of-the sample forecasting performance and also avoids the problem of over fitting.

The general expression for the final output value \( y \) in a multi-layer feed forward neural network is given by

\[
y = g \left( \sum_{j=0}^{q} \alpha_j f \left( \sum_{i=0}^{p} \beta_{ij} x_i \right) \right)
\]

Where \( f \) and \( g \) denote the activation function at hidden and output layer respectively, \( p \) is number of input nodes, \( q \) is the number of hidden nodes, \( \beta_{ij} \) is the weight attached to the connection between \( i^{th} \) input node to the \( j^{th} \) node of hidden layer, \( \alpha_i \) is the weight attached to the connection from \( j^{th} \) hidden node to the output node and \( x_i \) is the \( i^{th} \) input of the model. Each node of the hidden layer receives the weighted sum of all inputs including a bias term for which the value of input variable will always take a value one. This weighted sum of input variables is then transformed by each hidden node using the activation function \( f \) which is usually non-linear sigmoid function. In a similar fashion, the output node also receives the weighted sum of the output of all hidden nodes and produces an output by transforming the weighted sum using its activation function \( g \). In time series analysis, \( f \) is often chosen as logistic function and \( g \) as an identity function. For \( p \) input nodes, \( q \) hidden nodes, one output node and biases at both hidden and output layer, the total number of parameters (weights) in a three layer feed forward neural network is \( q(p + 2) + 1 \).
For a univariate time series forecasting problem, past observations of a given variable serves as the input variables. The neural network model attempt to map the following function

\[ y_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-p+1}, w) + \varepsilon_{t+1} \]

where \( y_{t+1} \) pertains to the observation at time \( t+1 \), \( p \) is the number of lagged observation, \( w \) is the vector of network weights, and \( \varepsilon_{t+1} \) is the error term at time \( t + 1 \). Hence, the neural network acts like nonlinear autoregressive model. Suppose a time series data contains \( N \) observations \( y_1, y_2, \ldots, y_N \) and out of \( N \) data points, \( n \) observations \( y_1, y_2, \ldots, y_n \) are available for training purposes and the model contains \( p \) lagged observations as input variable then \( n - p \) patterns will be available for training the network for one-step ahead forecasting. This implies that \( y_1, y_2, \ldots, y_p \) will serve as first input patterns for predicting the target output \( y_{p+1} \). The last training pattern will be \( y_{n-p}, y_{n-p+1}, \ldots, y_{n-1} \) for predicting the target output \( y_n \). Once the number of layers and total number of nodes in each layer has been determined, the network is ready for training, a parameter estimation process. The objective of training is minimization of an error function that measures the misfit between the predicted value and the actual value for any given value of \( w \). The error function which is widely used is given by the sum of the squares of the error between the predicted value \( \hat{y}_t \) for time \( t \) and the corresponding target value \( y_t \) at time \( t \), so that we minimize

\[ E(w) = \frac{1}{2} \sum_{t=p+1}^{n} (y_t - \hat{y}_t)^2 \]

where the factor 1/2 is included for mathematical simplification. The error surface for multilayer feed forward neural network with non-linear activation function is complex and believed to have many local and global minima (Patterson, 1996). This is the main reason for the development of several evolutionary computation optimization methods for neural network training.

**Particle Swarm Optimization for Neural Networks**

Particle Swarm Optimization (Kennedy & Eberhart, 1995) is a swarm intelligence technique inspired by the behavior of birds flocking. It is one of the two popular computational intelligence techniques. The other is ant colony optimization (ACO) inspired by ants foraging for food. PSO has been used in many applications such as neural network training (Chen, Yang & Dong, 2006), and fuzzy control systems (Abrahams & Khan, 2003).

Consider the scenario of birds looking for food. The birds perform a random search of the food in a certain area. The location of the food is not known. In each iteration, each bird first looks for the food within its own neighborhood and decides on a location which may lead to the food source. At the end of the iteration, the birds compare their locations and follow the bird that is closest to the food source. Therefore, in each iteration the birds are aware of how far the food source is from the current location. The "bird" or a particle is a solution in the search space. This idea is translated into PSO.

PSO begins by initializing a random number of particles in the search space. Each particle is treated as a point in an N-dimensional space. The total number of parameters \( q(p + 2) + 1 \) of the neural network model \((p,q,1)\) serves as the dimension for each particle in the search space. Each particle is associated with two parameters: velocity and position for each dimension. The particles fly through the
search space moving with a certain velocity and changing the position in the hope of getting closer to the target. There is a fitness function that is evaluated by each particle for its own fitness value. In this case error function has been used as a fitness function. The PSO travels in space based on two best values. The particle retains its previous best position for each dimension (fitness). This is called $p_{id}$ for $i^{th}$ particle of $d^{th}$ dimension. This $p_{id}$ value indicates the best solution the particle has achieved so far. The second value is the global best called as $p_{gd}$. This value is obtained by comparing the best values of all the particles for a particular dimension. Using these two values, $i^{th}$ particle updates its velocity and position for $d^{th}$ dimension as follows:

$$v_{id}(t) = \gamma \cdot v_{id}(t-1) + c_1 \cdot r_1 \cdot (p_{id} - x_{id}(t-1)) + c_2 \cdot r_2 \cdot (p_{gd} - x_{id}(t-1))$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t)$$

$v_{id}$ is the velocity of particle $i$ for the dimension $d$, $x_{id}$ is the position of particle $i$ for the dimension $d$, $c_1$, $c_2$ are the learning factors, $r_1$, $r_2$ are uniform random number in the range $[0,1]$ and $\gamma$ is the inertia weight. Several variations of the basic PSO algorithm has been developed, which improve the performance on practical problems. Trelea (Trelea, 2003) proposed an improved deterministic PSO algorithm. The expression for position and velocity update for particle $i$ of the dimension $d$ is given below:

$$v_{id}(t) = a \cdot v_{id}(t-1) + b \cdot (p_{id} - x_{id}(t-1)) + b \cdot (p_{gd} - x_{id}(t-1))$$

$$x_{id}(t) = c \cdot x_{id}(t-1) + d \cdot v_{id}(t)$$

After dynamic analysis and simulation experiments, Trelea emphasized two set of parameters for $a$ and $b$. Parameter set 1 refers to $a = 0.6$ and $b = 1.7$ and parameter set 2 refers $a = 0.729$ and $b = 1.494$. These are commonly known as Trelea I and II. The other two parameters $c$ and $d$ were set as 1 so that the variable $v$ can be interpreted as true velocity.

**Adaptive Neuro-Fuzzy Inference System (ANFIS)**

The adaptive neuro-fuzzy inference system (ANFIS) is a fuzzy inference system implemented in the framework of adaptive networks. In ANFIS, membership function parameters are adjusted using a hybrid learning procedure involving back-propagation algorithm and least squares with the help of given input-output data set instead of experience or intuition (Jang, 1997). Jang introduced an ANFIS architecture, which is a five layer feed-forward neural network with supervised learning capability. The working of different layer of this network (Figure 1) is given below. Let $x (y)$ represents input variable, $A_{i}(B_{i})$ correspond to linguistic label and $O_{j,i}$ denotes the output of $i^{th} (i = 1, 2)$ node in $j^{th} (j = 1, 2, ...,5)$ layer.
First layer: Selection of input variable and fuzzification through membership function is done in the first layer. All nodes of this layer are adaptive. The parameters of a node can change the shape of the membership function. In this study generalized bell function is chosen as a membership function which has three parameters \( \{a_i, b_i, c_i\} \) referred as premise parameters.

\[
O_{1,i} = \mu_{A_i}(x) = \frac{1}{1 + \left( \frac{x - c_i}{a_i} \right)^2 b_i} 
\]

Second layer: Each node of this layer is fixed one. Each node of this layer computes the firing strength of each rule by multiplying the all incoming signals.

\[
O_{2,i} = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y) 
\]

Third layer: Each node of this layer is also fixed one. The output of each node is relative contribution of each rule. Outputs of this layer are also known as normalized firing strength.

\[
O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2} 
\]
Fourth layer: Every node in this layer is an adaptive one. The output of each node of this layer is the product of relative contribution of each rule and the fuzzy if-then rules (in this case a first order Sugeno fuzzy model). An example of first order Sugeno fuzzy model is: if \( x \) is \( A_1 \) and \( y \) is \( B_1 \), then \( f_1 = p_1 x + q_1 y + r_1 \).

\[
O_{4, i} = w_i f_i = w_i (p_i x + q_i y + r_i)
\]

The parameter set \( \{p_i, q_i, r_i\} \) of this layer are referred as consequent parameters.

Fifth layer: The single node of this layer calculates the final output by summing over all the incoming signals.

\[
O_{5, 1} = \sum_i w_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}
\]

The structure of ANFIS is not unique. In the extreme case, the whole network can be reduced to a single adaptive node with the same parameter set.

Empirical Results

Data and Implementation

In order to empirically investigate the potential of ANFIS, PSO-based neural network and a standard neural network model for time series forecasting, we applied it to a real financial time series data. We used the daily closing price data of Standard and Poor's 500 index (S&P 500 index) obtained from Yahoo! finance. The sample data covered the time period from 2 January 2004 to 31 December 2007 and contains a total of 1006 observations. We divided the data series into three parts: training set, validation set and testing set for conducting the experiment in order to determine the best ANFIS and neural network structure. The training set is presented for parameter estimation, validation set is used to measure network generalization and testing set provides out of the sample performance. Major portion of the data is used for training and the rest of the data is divided into validation and testing set. In this study, all ANFIS and neural network modelling was done using fuzzy and neural network toolbox of MATLAB while the PSO related algorithm was implemented employing the PSO toolbox developed by Birge (Birge, 2003). This toolbox is available at central file exchange of Matlab with five types of PSO (Common PSO with and without inertia, Clerc's constricted type, Trelea 1 and Trelea 2).

An ANFIS model design requires the specification of input variables, number and type of membership function for input variables and type of membership function for the output variable. In case of time series, different lags and various order of moving average of the series serve as input variables. For this study we tried different combinations of lags, levels of moving averages and different numbers and types of membership functions to get the presented results. The best model was obtained with four inputs and two generalized bell membership functions for each input variable. The linear membership function was selected for output variable which means first order Sugeno-type model. The generated fuzzy inference system consists of 16 fuzzy rules \( 2^4 \) with a total of 124 parameters \( 24 \) premise parameters and 80 consequent parameters). The generated model was trained with a hybrid algorithm that uses backpropagation for hidden layer and least squares for output layer.

A three layer feed forward neural network model has been used for this study. Following previous studies (Boyd, 1996; Qi & Zhang, 2001) the logistic and identity function have been used as activation
function for the hidden nodes and output node respectively. Since multi-step ahead forecasting has been done using iterative procedure so only one output node is employed. Hence, the model uncertainty is associated only with the number of input nodes ($P$) which is the number of lagged observations in this case and number of hidden layer nodes ($q$). The number of input nodes and hidden nodes were determined with the help of experimentation. We varied the number of input units from 1 to 10 as it plays a significant role in mapping the autocorrelation structure. The number of hidden units varied from 4 to 10 with an increment of 2. Thus, a total of 40 neural network models were tried before arriving at the final structure of the model. There are many variations of the backpropagation algorithm used for training feedforward networks. In this study, the Levenberg-Marquardt algorithm which has been designed to approach second-order training speed without computing the Hessian matrix has been employed. It has been shown (Demuth & Beale, 2002) that this algorithm provides the fastest convergence for moderately sized feedforward neural network used on function approximations problems.

Particle swarm optimisation is a population based search technique, that finds the best solution in the search space. As mentioned earlier, different variation of basic PSO algorithm has been developed and the best choice depends on the objective function being optimized. The error surfaces for neural networks are much irregular and contain several irregular geometries. Hence, for this study both Trelea I ($a = 0.6, b = 1.7$) and Trelea II ($a = 0.729, b = 1.494$) (Trelea, 2003) have been used for training the neural network. The value of $c$ and $d$ were set to 1 as mentioned above. The parameter $a$ represents inertia term. The inertia term plays an important role on the performance of PSO as it enables balance between local and global search during optimization process (Shi & Eberhart, 1998). Another important characteristics of this optimization process is acceleration constant $b$. Acceleration constants represent relative impact of personal best and global best positions on the velocity of a particle. In addition to parameters $a$ and $b$, the performance of algorithm is influenced by the number of particle in the swarm. It has been shown (Trelea, 2003) that a medium number of particles provide best results. Further, the number of particles does not improve the problem solving ability of the algorithm once the number is more than a minimum which ranges from 24 to 30. Therefore, a swarm size of 30 particles has been used for this study. As indicated earlier, each particle is associated with two parameters: position and velocity. Initially, we randomly distribute particles in several nodes of the search space. Each data point is associated with a location and a velocity. All these nodes in the domain represent a neural network model. The maximum particle velocity is set as $(d_{\text{max}} - d_{\text{min}})/2$ where $d_{\text{max}}$ and $d_{\text{min}}$ are the maximum and minimum values for the $d^\text{R}$ dimension. The mean squared error has been used as a fitness function. Each particle has 61 dimensions as a neural network with eight input nodes and six hidden nodes was selected on the basis of test performance. The global version (all particles being neighbours of each other) of the algorithm has been used. The optimization experiment with all the three training algorithms was run 50 times with different initial values for position and velocity in order to record the best value for performance measures.

The forecasting ability of all the four models has been assessed with respect to two common performance measures viz. the mean squared error and the mean absolute deviation. The mean square error measures the overall performance of a model and is given by

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

where $y_i$ is the actual observation at time $i$ and $\hat{y}_i$ is the predicted value for time $i$ and $n$ is the number of predictions. The mean absolute error is a measure of average error for each point forecast and is given by
where the symbols have the same meaning as above. The prediction performance for the ANFIS model, standard back propagation and PSO based neural network models have been evaluated for three forecast horizons that is ten days ahead, thirty days ahead, and sixty days ahead.

**Discussion**

The first and most important step in any time series analysis is to plot the data. Following (Meese & Rogoff, 1983) we applied natural logarithmic transformation to the data to stabilize the series and has been plotted in Figure 2. We note from the graph that the series is non-stationary and shows a irregular upward trend, hence some sort of differencing is required. Accordingly, first differencing of the data was done and plotted in Figure 3 which indicates that the differenced series is stationary. This was further confirmed from the plot of sample autocorrelation function of the series.

**Figure 2**

**Log(S&P 500)**
The best ANFIS model was obtained with four inputs and two generalized bell membership functions for each input variable. A neural network model with eight input nodes and six hidden nodes provided the best test result. In order to eliminate the effect of initial value and to increase the possibility of obtaining the true global minima we trained each neural network model 20 times by using different initial weights to select the best model for prediction using the given data set. Table1 present the comparison of out of the sample prediction performance for all the four models viz. Adaptive neuro-fuzzy inference system (ANFIS), Levenberg-Marquardt algorithm based neural network (SNN), neural network based on Trelea I (NN-TreleaI) and Trelea II (NN-TreleaII) for all the three forecast horizons namely 10 days ahead, 30 days ahead and 60 days ahead respectively.

Results clearly show that both PSO based training algorithms provide better forecasting ability with respect to the MSE and MAE across all the three forecasting horizons in comparison to ANFIS and standard neural network. For the present data set the performance of ANFIS model is poor in comparison to simple neural network based model. Implicit use of fuzzy logic did not provide much help in modelling the volatility of financial seires. The performance of ANFIS model depend on the distribution of the training and test data set. Better performance could be expected if they have similar distribution which can be achieved using other scheme of the data division. In this case test data consists of last part of series which exhibit more volatility than the earlier part of the series which is being used for training the model. Further, neural network model based on Trelea II uniformly provided better forecasting accuracy than Trelea I with respect to performance measures across all the three forecasting horizons except for the 60 days period with the MAE. The prediction ability of all the training algorithms deteriorated with the increase in forecasting periods which is obvious. Because of this reason neural network based models are preferred for short term forecasting.
Table 1

Comparison of Out of the Sample Prediction Performance for Three Horizons

<table>
<thead>
<tr>
<th>Model</th>
<th>10 days MSE</th>
<th>10 days MAE</th>
<th>30 days MSE</th>
<th>30 days MAE</th>
<th>60 days MSE</th>
<th>60 days MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFIS</td>
<td>5.8329</td>
<td>5.3126</td>
<td>16.2214</td>
<td>9.3786</td>
<td>18.6491</td>
<td>13.7634</td>
</tr>
<tr>
<td>SNN</td>
<td>4.4547</td>
<td>5.4954</td>
<td>14.6010</td>
<td>8.2843</td>
<td>17.6020</td>
<td>10.1930</td>
</tr>
<tr>
<td>NN-TreleaI</td>
<td>0.5935</td>
<td>1.9489</td>
<td>7.1855</td>
<td>6.4867</td>
<td>9.9665</td>
<td>7.9468</td>
</tr>
<tr>
<td>NN-TreleaII</td>
<td>0.3346</td>
<td>1.2446</td>
<td>6.3307</td>
<td>5.5530</td>
<td>9.7403</td>
<td>8.3913</td>
</tr>
</tbody>
</table>

Note: a) All MSE values should be multiplied by $10^{-5}$ and
b) All MAE values should be multiplied by $10^{-3}$.

Conclusions

Artificial intelligence models are a promising alternative to the traditional time series models. In this study we tried to examine whether implicit use of fuzzy logic with neural network can improve the prediction ability of neural network with the help of a real financial time series data. Adaptive neuro fuzzy inference system (ANFIS) combines the fuzzy qualitative approach with the neural networks adaptive capabilities to model input output relationship. Efforts were also made to assess the prediction performance of particle swarm optimization trained neural network.

Results showed that PSO based training algorithm provided better forecasting ability with respect to the MSE and MAE across all the three forecasting horizons. Although ANFIS have been proved to be more successful than standard neural network, it was not found true with the data used for this study. Further, Trelea II provided better performance than Trelea I with respect to different performance measures. Needless to mention, this study is limited in the sense that experiment has been carried out with one set of data.
References


